2022

2º Trabalho

Modelação e Programação

HERANÇA, CLASSES ABSTRACTAS E INTERFACES

DATA DE ENTREGA: 17/04/2022

Este trabalho tem por objetivo a familiarização com a herança de classes (e classes abstratas e interfaces). O código deste trabalho deve figurar no package *tp2*.

Importante: Este trabalho está isento de relatório se o código entregue estiver devidamente comentado seguindo as regras semânticas da ferramenta Javadoc (os outputs destas ferramenta deverão ser entregues conjuntamente com o código fonte do trabalho).

Parte A – Colecções com colecções, usando herança

Colocar no package *tp2.pack1ColecoesHeranca* as classes deste cenário.

Pretende-se que os alunos utilizem a herança (classes abstratas e interfaces) e que verifiquem as suas propriedades e potencialidades. O contexto utilizado é o cenário do TP1, mas agora os livros e as colecções são classes derivadas de classe base **Obra**. As colecções continuam a suportar livros e outras colecções dentro delas, e cada colecção não deve permitir a existência de obras, diretamente dentro dela, com nomes repetidos. Deve-se começar por colocar os ficheiros disponibilizados na sua área de trabalhos (**tp2**) e depois editar os ficheiros **Obra**, **Livro** e só depois **Coleccao**, copiando ou adaptando os conteúdos análogos e já concebidos no TP1, mas tendo em conta o novo cenário, o UML mostrado na Figura 1, as descrições anexadas e os outputs desejados.

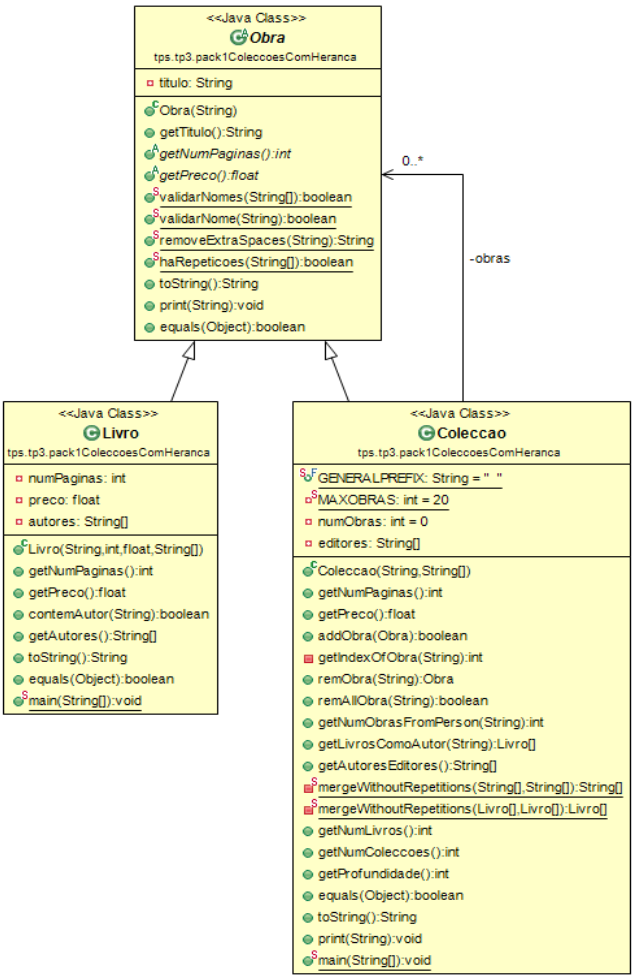


Figura 1 - Classes para suporte a Coleções

Deve-se consultar: os comentários existentes nos ficheiros disponibilizados; os seus *mains*; e os respetivos *outputs* desejados. Salienta-se o seguinte para cada uma das seguintes classes:

Obra, Livro e **Coleccao**:

Os métodos ***toString***, ***equals*** *e* **print** devem ser concebidos tendo em conta o código das respetivas sobreposições e os outputs desejados.

Livro:

O array autores, assim como o array de editores em Coleccao, devem ficar com a dimensão correta, de forma a não conterem entradas a *null*.

Coleccao:

Esta classe só deve possuir um array de Obra, que deverá conter os livros e as coleções. Este array deve ser gerido de forma às suas obras ficarem sempre nos menores índices. Agora os livros irão ficar misturados com as coleções.

O método **remAllObra** remove todas as obras com o título igual ao título recebido, ignorando as diferenças entre maiúsculas e minúsculas (*ignore case*), mesmo que estas obras se encontrem dentro de subcolecções e em qualquer nível de profundidade. Devolve *true* caso tenha removido pelo menos uma obra.

O método **getProfundidade** deverá devolver o comprimento máximo das cadeias de coleções com coleções dentro. Uma colecção, por si, deve devolver uma profundidade de 1.

No exemplo c1[c2[c3[]] c4[]]: c1 deverá devolver 3; c2 devolver 2; e c3 e c4 devolver 1.

Interfaces:

Extraia-se a interface da classe Obra, com o nome *IObra* e utilize-a em todos os locais possíveis do código. Utilize a funcionalidade de refactoring (Refactor – Extract interface…) do IDE para tal efeito (tal como indicado nos slides). Remova os métodos desnecessários em Obra.

Deve-se completar os métodos "main" de forma cobrir todas as possíveis situações de utilização das classes e métodos existentes.

Parte B – Festivais

Colocar no package *tp2.pack2Festivais* as classes deste cenário.

Neste grupo pretende-se um sistema, para uma agência de eventos, que modele **espetáculos** e **festivais**. Os espetáculos devem ter um ou mais artistas, em que cada artista só pode atuar uma vez em cada espetáculo. Os **festivais** podem conter espetáculos e outros festivais.

Tendo em conta o diagrama de classes mostrado na Figura 2 - Classes para suporte a Festivais, implemente a totalidade das classes: **Evento**; **Espetáculo**; e **Festival**.

Implemente a classe abstrata **Evento**, tendo em conta que: **getNumBilhetes**, **getArtista**, **numActuacoes** são métodos abstratos; e **toString** deve devolver "NoitadaAzul com 2000 bilhetes e com os artistas [Joana, Artur]", para um evento de nome NoitadaAzul com 2000 bilhetes e com os artistas enunciados. A palavra artistas pode aparecer sempre no plural.

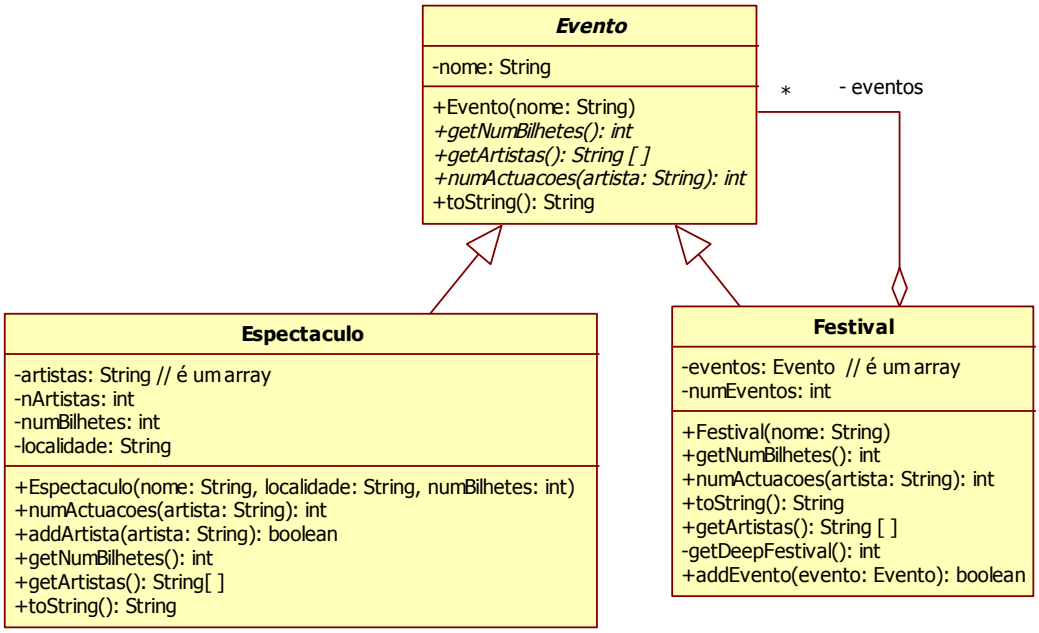


Figura 2 - Classes para suporte a Festivais

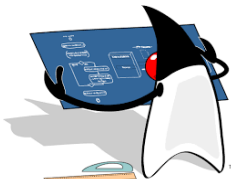
Implemente a classe **Espetáculo**, tendo em conta que: **construtor**, deverá ter a capacidade para um máximo de 10 artistas; **numActuações**, devolve o número de atuações do artista recebido (um ou zero); **addArtista**, adiciona o artista, caso ele já não esteja registado no espetáculo; **getNumBilhetes**, devolve o número de bilhetes disponíveis no espetáculo; **getArtistas**, deve devolver um novo array só com os artistas registados e com essa dimensão; e **toString**, deve devolver um conteúdo idêntico ao toString de Evento seguido de " em Lisboa", para o caso em que o espetáculo se realize na localidade de lisboa.

Implemente a classe **Festival**, tendo em conta que: **construtor**, cada festival deverá ter uma capacidade para um máximo de 20 eventos, que podem ser Espetáculos ou Festivais; **getNumBilhetes**, devolve o número de bilhetes disponíveis da totalidade dos eventos do festival; **numActuações**, devolve o número de atuações, dentro do Festival, do artista recebido; **toString**, deve devolver "Festival " seguido do conteúdo idêntico ao toString de Evento; **getArtistas**, deve devolver um novo array só com os artistas registados e com essa dimensão, cada artista só deve aparecer uma vez no array devolvido, sugere-se a utilização de uma List<String> (e utilizar contains e toArray(…)); **getDeepFestival**, devolve a profundidade máxima de festivais dentro de festivais do festival corrente (o próprio festival não conta); e **addEvento**, adiciona o evento, caso para nenhum dos artistas do novo evento se verifique que o seu número de atuações no novo evento (a adicionar) supere em mais de duas o número de atuações no festival corrente. Implemente também um novo método, que não consta do UML, *boolean* ***delEvento****(String nomeEvento)*, que deverá remover o evento, com o nome recebido, em qualquer profundidade do Festival corrente e devolver *true* se removeu. O **array de eventos** deve ser gerido tal que nas remoções não se

realizem deslocamentos (*shifts*) dos eventos existentes, ou seja, o array poderá conter nulls entre elementos.

As classes Espetáculo e Festival deverão ter um método main que deverá testar todos os métodos da classe.

O código deste trabalho deve ser entregue até ao dia **17 de abril** de 2022.



Bom trabalho, Carlos Júnior, João Ventura, Pedro Fazenda